Amendments to the Specification:

Please replace paragraphs [0033]-[0034] as follows:

[0033] FIG. 1 is a Use Case Map showing policy execution on call control, as is known in the art; ~~and~~

[0034] Figure 2 is a Use Case Map showing policy management, conflict detection, and conflict resolution according to the present invention; [.]

After paragraph [0034] add the following paragraphs, [0034.1] – [0034.4]:

[0034.1] Figure 3 is a block diagram showing a method of user policy management;

[0034.2] Figure 4 is a block diagram of a step of receiving user-entered policies in the method of Figure 1, according to an exemplary embodiment;

[0034.3] Figure 5 is a block diagram showing translation of an exemplary user-entered policy into executable feature language as scripts representing branches of a decision tree, and translation of the policy into a policy language, according to an exemplary embodiment; and

[0034.4] Figure 6 is a block diagram showing an example of errors that are common to naïve users.

Please replace paragraph [0072] as follows:

[0072] The system of FIG. 2 provides an overall mechanism for the creation, management, testing and provisioning of policies as depicted in the flowchart of FIG. 3. More particularly, the method according to an exemplary embodiment includes: entering user policies (300); translating this user-understandable language into an executable feature language (305); translating the features from executable

feature language into a policy language (310), such as FIAT, from which it is possible to detect common feature specification errors; analyzing the errors (315) in a manner that is aware of the expectations and common errors of naïve users; report these errors (320) to the user (e.g. via the Web interface) in terms that are understandable to naive users and compatible with how the policies were originally described; provide the user with options to either accept the interactions as they are, repair them manually or to accept a recommendation of an automatic correction(325); and uploading the policies for execution (330). Through the Web interface (with text or voice browser) a user may enter a policy in a user friendly and intuitive way. This may be done by filling in structured sentences or by filling in forms, via a Web interface tailored to the particular environment of the user (hospital, law firm, store, school, home, etc.). The system of the invention provides structured information that may be converted into a formal representation (e.g. CPL or FIAT).

Please replace paragraphs [0074] – [0083], as follows:

[0074] The Web interface enables a user to manage his/her list of policies (e.g. in a list box or in a different panel or frame), typically sorted by name or by priority. The following operations on policies are supported (via buttons, links, menus, or voice activation), as shown in FIG. 4:

[0075] Create (400): Create a new policy, add it to the list, and activate it.

[0076] Modify (410): Modify the selected policy.

[0077] Delete (420): Delete the selected policy from the list

[0078] Duplicate (420): Make a copy of the selected policy (with the intention of modifying it later).

[0079] Deactivate (440): Deactivate the selected policy. Policies that are inactive are still kept on the list but are marked as such (e.g. different color or shadowed text). They are not used for validation or execution, but they are kept for future activation.

[0080] Activate (450): Activate the selected inactive policy.

[0081] Set Priority (460): Set the priority of the selected policy, which can be an absolute priority or a relative one (e.g. move the policy higher or lower in the list when sorted by priorities).

[0082] Validate (470): Detect and report conflicts in the list of active policies.

[0083] Approve (480): Approve the current list of policies and enable them for execution (e.g. through the generation of a single CPL script uploaded to the call control switch 1 or to the policy agent 3).

Please replace paragraph [0115], as follows:

[0115] For example, as depicted in FIG. 5, the following is a policy (500) that forwards all incoming calls to Jim Darling, unless the call originates from Reception:

Please replace paragraph [0117], as follows:

[0117] As discussed above, the policy (500) is translated into an executable feature language (510) in the form of CPL scripts (515 and 520), as follows ~~generated from this policy are the following~~:

Please replace paragraphs [0119] and [0120], as follows:

[0119] It will be noted that the absence of specific actions to be taken in the first CPL script results in the default behaviour of the system, namely to accept the incoming call as depicted by the leaf at the bottom of the left branch of the decision tree 530, discussed in greater detail below.

[0120] The translation of the foregoing CPL scripts into FIAT policy language (540) becomes (550) and (560), respectively:

Please replace paragraph [0128], as follows:

[0128] The general structure of a CPL specification is a decision tree (530). At some point, an element of decision is elected (e.g. caller): either the condition is verified, the condition is not verified, or the element of decision is not present. In

each case, a corresponding sub-tree follows with further instructions. Eventually, each branch develops, not into sub-trees, but rather into a leaf that consists of an action to be performed (e.g. redirect, reject).

Please replace paragraphs [0132] – [0137], as follows:

[0132] The incoherences reported by FIAT contain information identifying the features, their priorities, and the incoherence itself. The first step toward the interpretation of the incoherence consists in determining the type of policy, namely whether they are general or specialized (335). In some cases, further understanding of the problem is obtained by comparing the relative priorities of the policies (340). Then, the problem is reported to the user. This reporting starts by identifying the category of incoherence (345), the role of each policy in the problem is exposed (350), an example of the possible misbehaviour resulting from the presence of the two policies is given (355) and, when applicable, one or many ways to correct the situation are proposed.

[0133] The following types of problems are reported to the user, as depicted in FIG. 6:

[0134] 0. Redundancy (600): Two general policies are active. A special case is Conflict with Redundancy (605), where a general policy and an exception for the other general policy lead to different resulting actions.

[0135] 1. Shadowing (610): A general policy (i.e. that applies to all members of a user-defined domain, but potentially with exceptions) overrides a specific policy (that applies to an enumeration of one or many individuals). The specific policy can never be triggered.

[0136] 2. Specialization (615): Notifies that a specific policy will be selected over a general policy of lower priority.

[0137] 3. Conflict (620): Two policies address the same situation (their preconditions overlap) but with different resulting actions.

Please replace paragraphs [0139] – [0143], as follows:

[0139] a. Edit a policy (3600, which enables the user to modify one of the conflicting policies.

[0140] b. Disable a policy (365), which deactivate a policy, without deleting it.

[0141] c. Set the priority of a first policy above/below the priority of a second policy 370).

[0142] d. Add an exception to a general rule (375).

[0143] e. Tolerate the conflict (380) and no longer report it (leave it to the system to decide).